ROS Path Planning

Wego & Industrial Robot

WeGo

목차

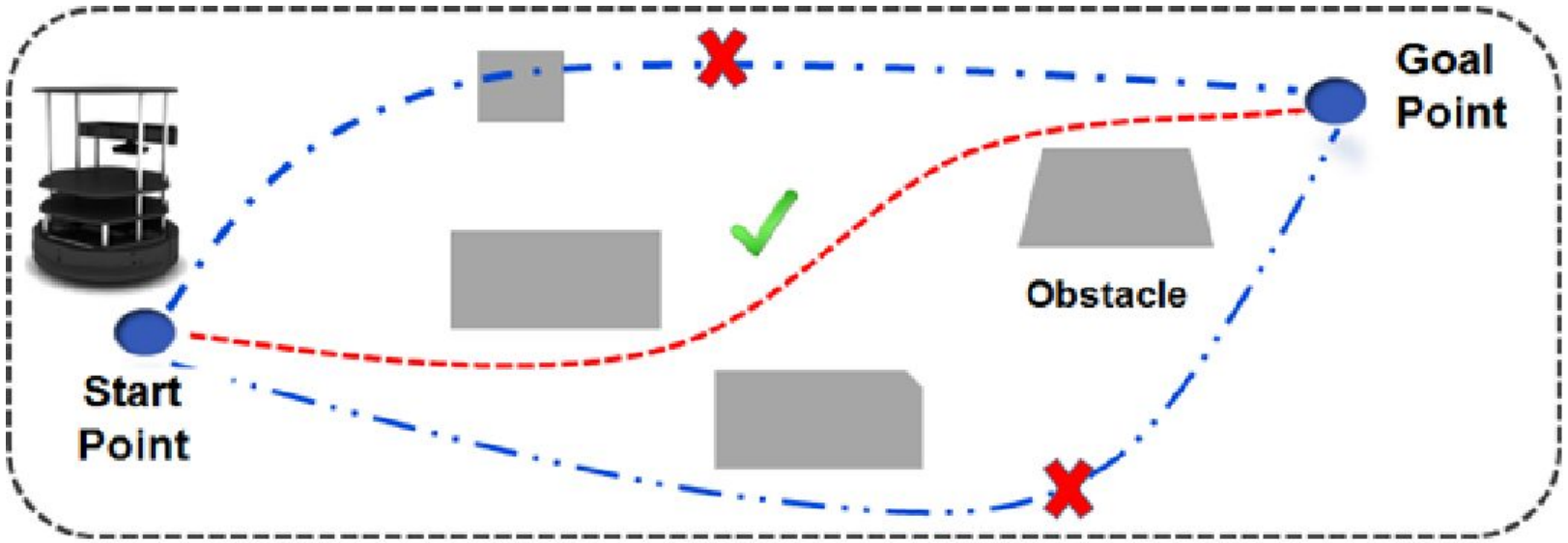
WeGo

01

Introduction to Path Planning

WeGo

- Path Planning
- Path Planning은 지도 상의 목적지가 주어졌을 때, 로봇의 형태와 상황을 고려하여, 목적지까지 이동하는 경로를 생성하는 기술
- 로봇에 한정되어 사용되는 것이 아닌 다양한 상황에서 경로 찾기 및 모션 계획에 사용된다.



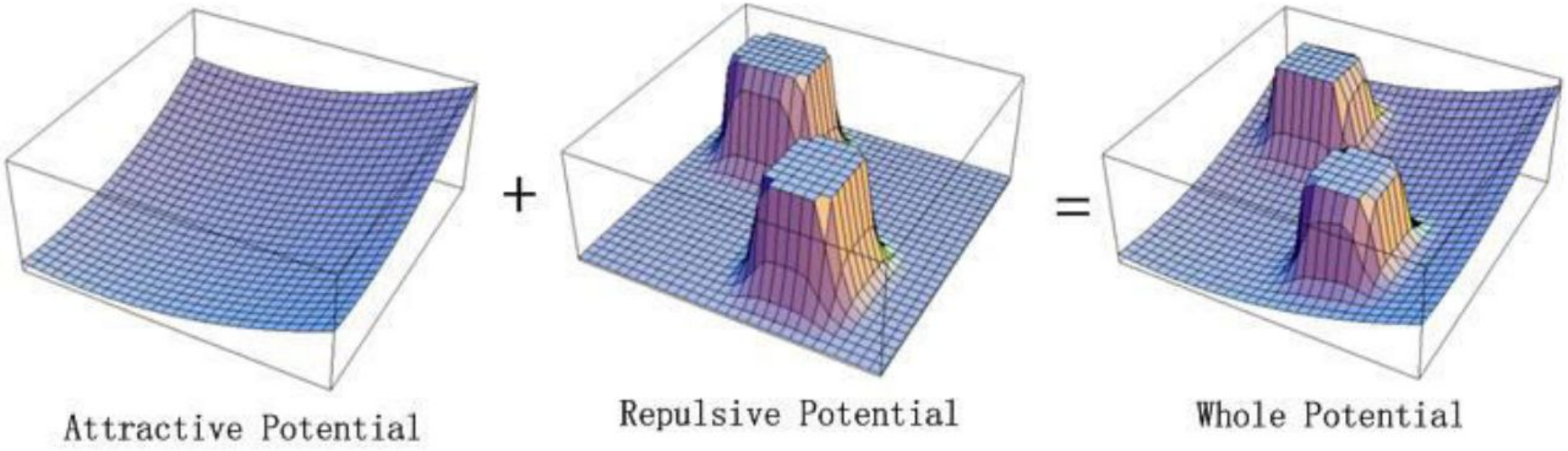
01 Introduction to Path Planning

- Path Planning
- 크게 나누면 Global Planner와 Local Planner로 구분할 수 있다.
- 전체 지도에서 Localization을 통해 얻은 현재 위치와 목표로 하는 목표 지점 두 가지 정보를 기반으로 하여, 지도 상에 있는 장애물을 회피하여 도착하는 경로를 생성하는 기술을 Global Planner라고 한다.
- 필요한 내용은 지도, 지도 상의 출발 위치, 지도 상의 도착 위치 세 가지 정보를 기반으로 하며, 현재 로봇이 측정하고 있는 주변 환경과는 무관하게 동작한다.

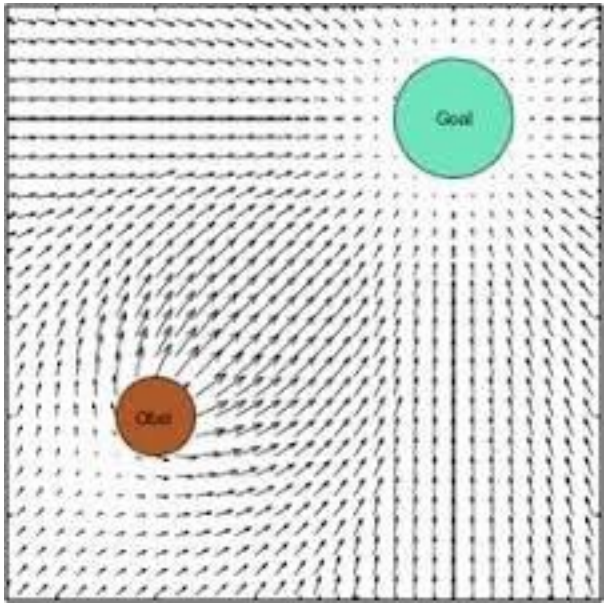
- Path Planning
- 반면 Local Planner의 경우, 입력 값으로 현재 로봇의 State, Sensor Data, Global Planner가 생성한 Global Path를 입력으로 받는다.
- 위 데이터를 기반으로 현재 위치에서 Global Path를 쫓아가기에 가장 최적의 Local Path를 생성한다.
- 서울에서 용인가는 경로 생성
- 용인 서울 고속도로 및 특정 국도를 이용하여 이동 (Global Planner)
- 3차선에서 1차선으로 차선을 변경하고 좌회전 (Local Planner)

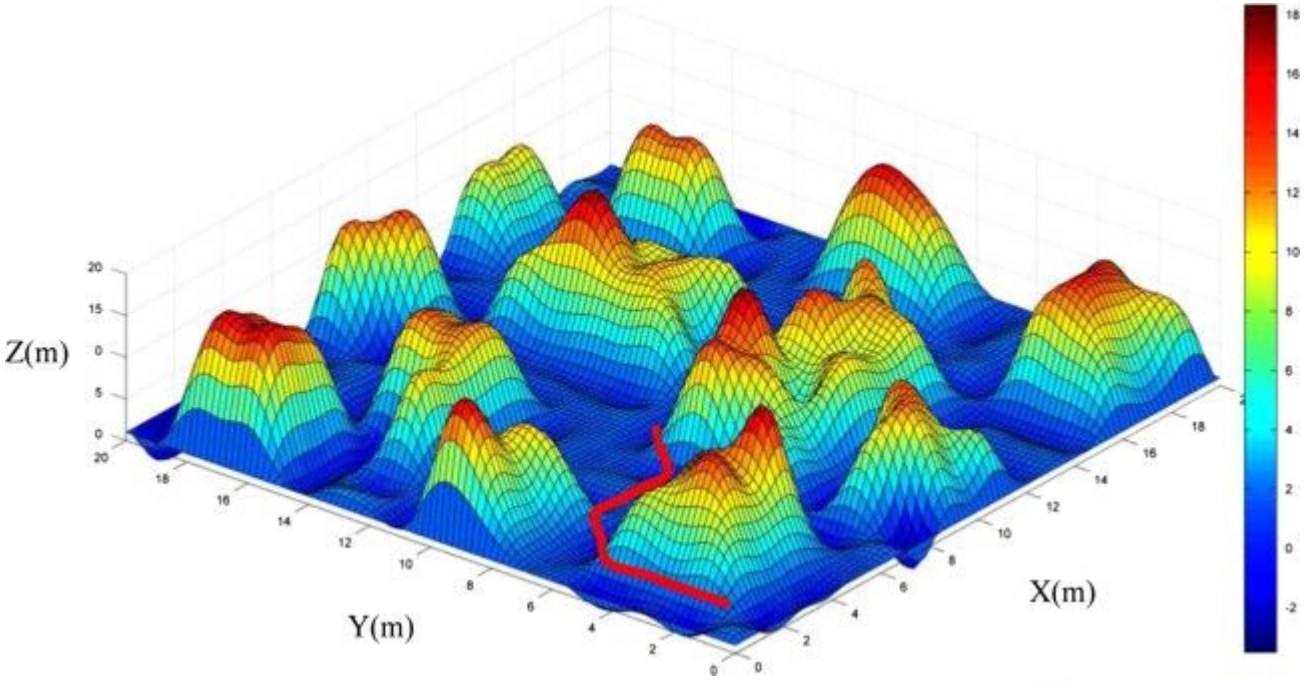
- Path Planning
- 여기서는 Grid 기반의 Path Planning을 의미하며, 이전에 생성한 Occupancy Grid Map을 기반으로 생성하는 Path를 의미합니다.
- Grid 기반의 Planning 방법은 크게 Potential Field 방식, Graph 기반 Search 알고리즘, RRT 등을 활용할 수 있습니다.

- Potential Field
- 출발 지점의 Potential을 가장 높게, 도착 지점의 Potential을 가장 낮게 설정한다.
- 중간에 있는 장애물의 위치에 대해서도 Potential을 Gaussian 함수와 같은 형태로 생성한다.
- 위 두 개의 Potential을 합쳐서 전체 Potential Field Map을 생성하고, 각 위치에서의 전체 Map에서의 Gradient를 계산하여, 이를 연결하는 방법을 통해 경로를 생성



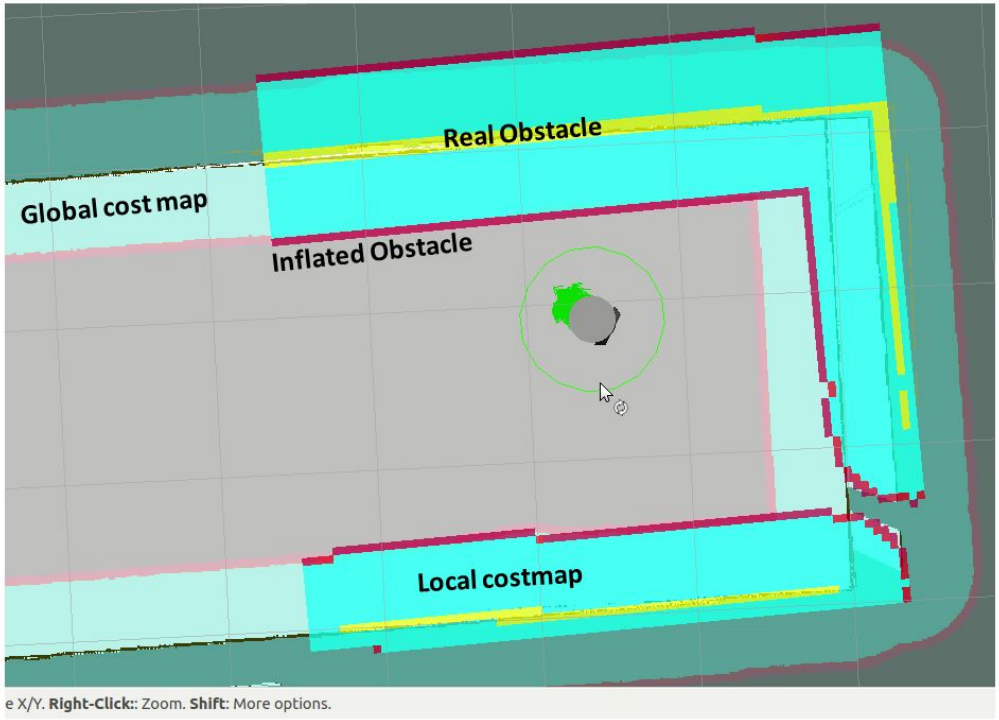
- Potential Field
- Gradient Descent 방식을 통해서 경로를 탐색
- Gradient를 이용하는 방식이므로, Local Minima, Saddle Point 등으로 인한 문제가 발생할 수 있음



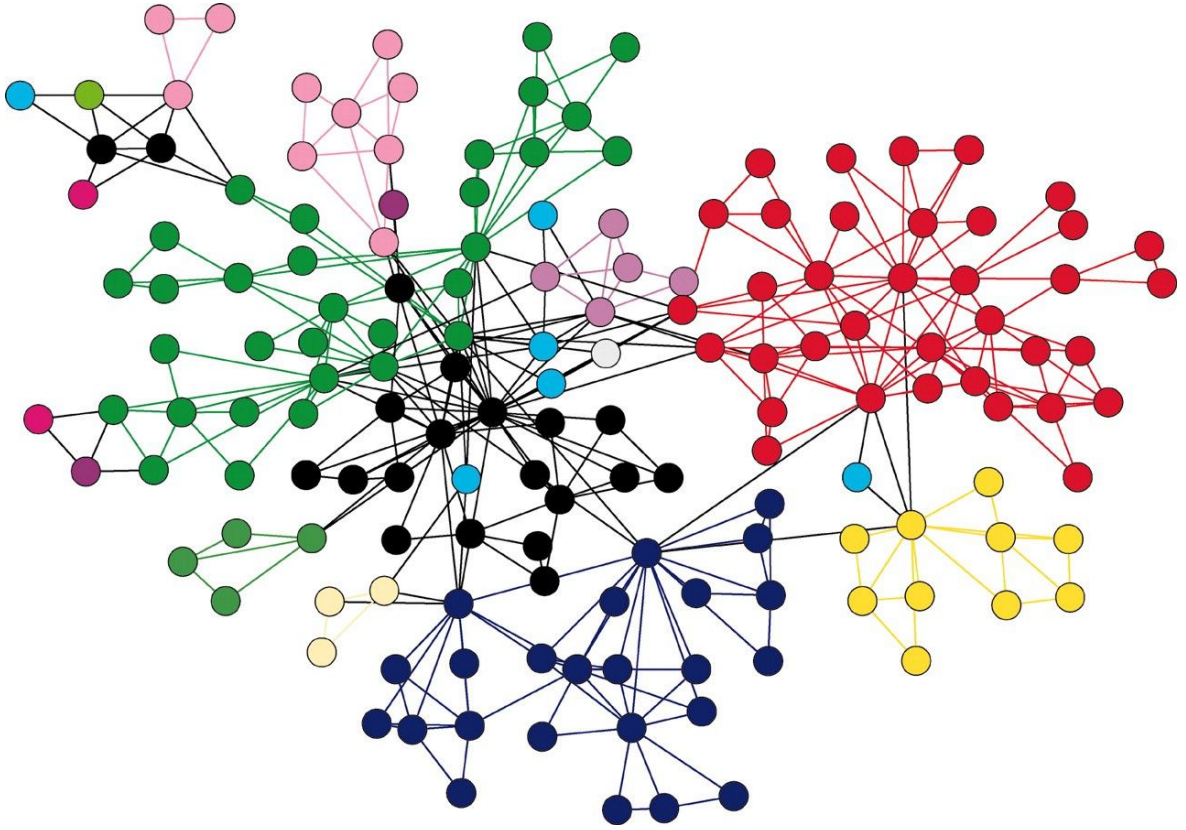


WeGo

- Graph-Based Search Algorithm
- 대표적으로 Dijkstra's algorithm, A*, Breadth-First Search, Depth-First Search 등이 있다.
- 우선적으로 Obstacle Growing을 적용해야하며, Growing Region의 경우, 이동하는 로봇을 원형으로 가정하였을 때, 최소 원의 반지름 이상이 되어야한다.

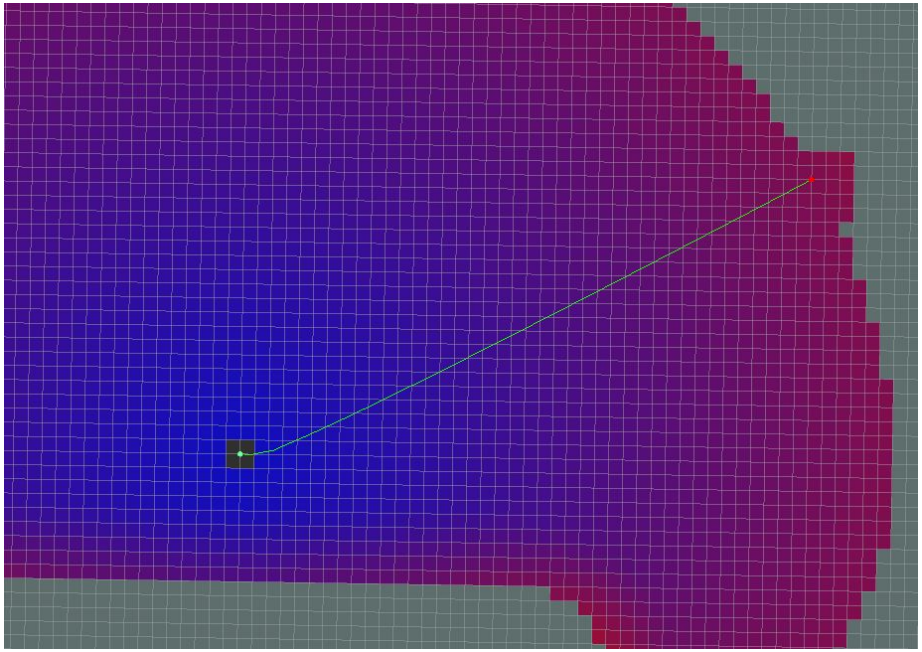


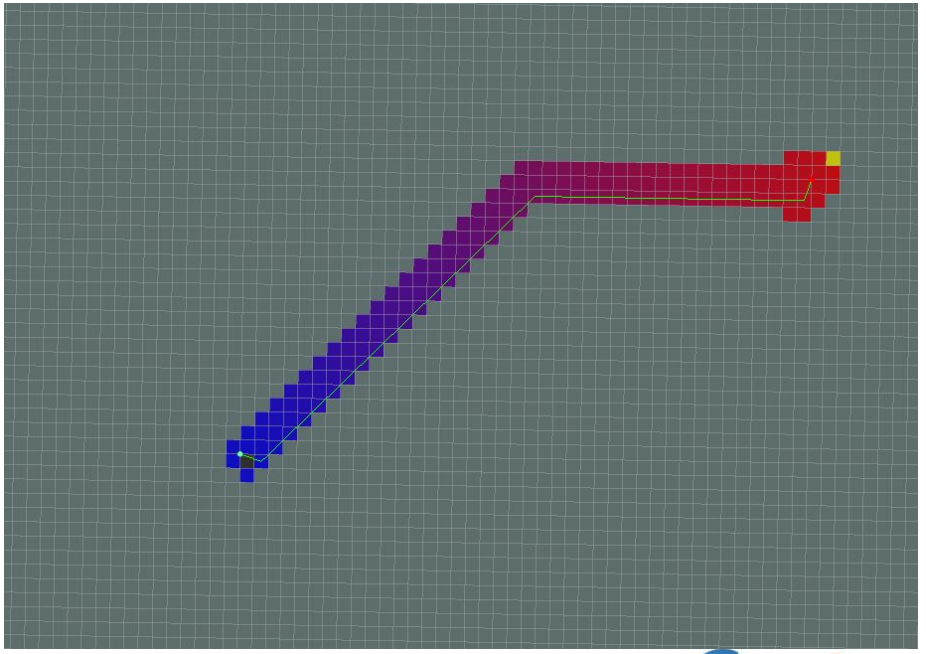
- Graph-Based Search Algorithm
- Growing 된 결과를 이용하여, 이를 Graph로 변환한다.
- Graph 변환시에는 4-neighbor인지 8-neighbor인지를 확인하여 변환해야한다.



WeGo

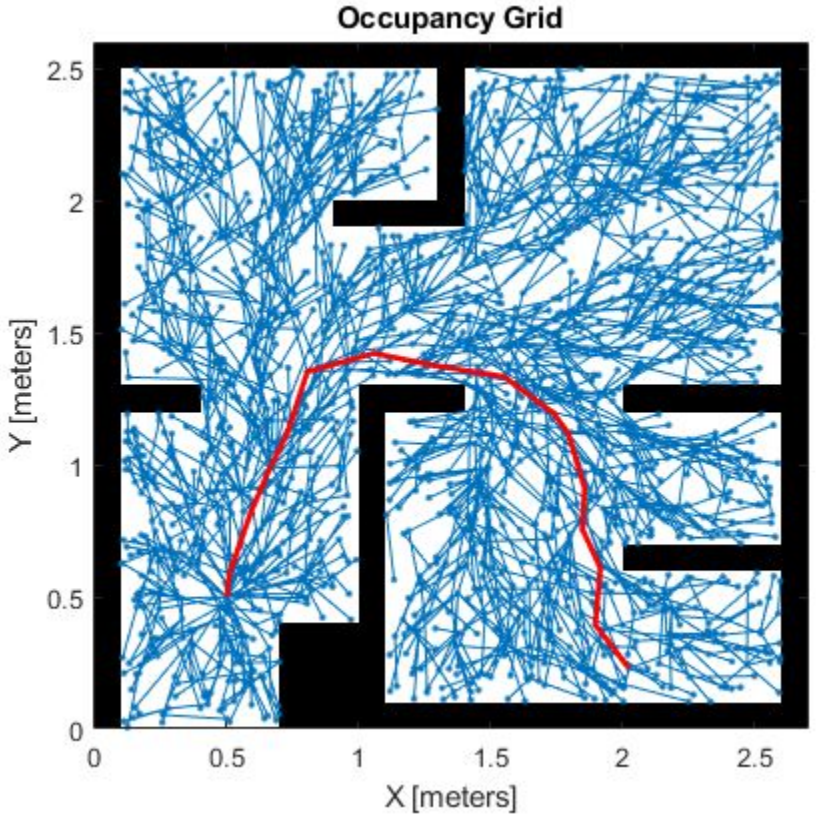
- Graph-Based Search Algorithm
- Growing 된 결과를 이용하여, 이를 Graph로 변환한다.
- Graph 변환시에는 4-neighbor인지 8-neighbor인지를 확인하여 변환해야한다.
- 변환 후, 시작 장소와 목표 장소를 입력으로 받은 후, 이를 기반으로 Graph-based Search Algorithm을 적용한다.





WeGo

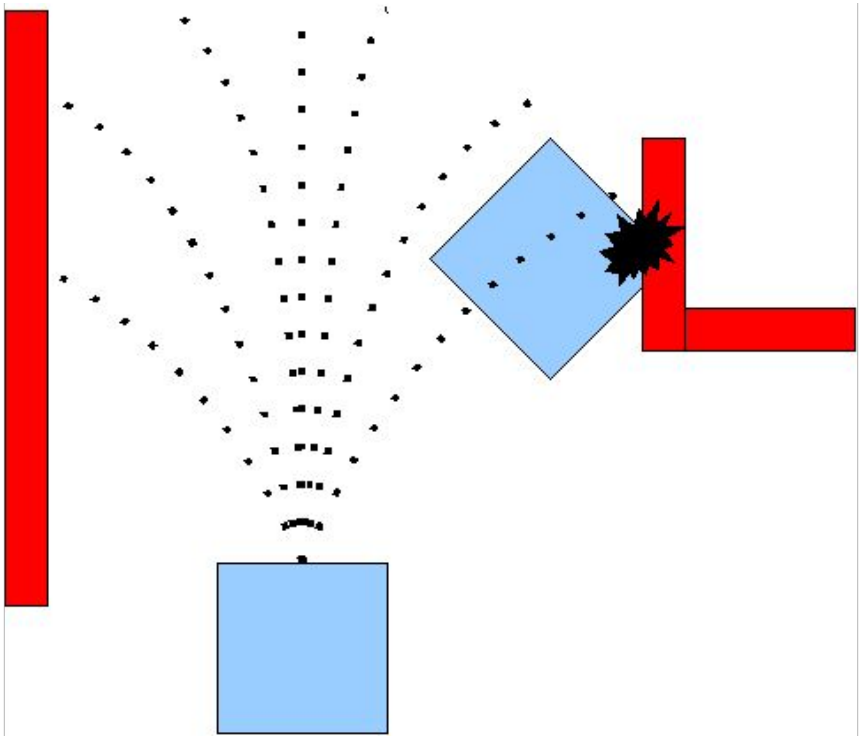
- RRT (Rapidly-Exploring Random Tree)
- 시작 장소로부터 Random Tree를 생성하여, 목표 장소에 도착하는 경로를 생성

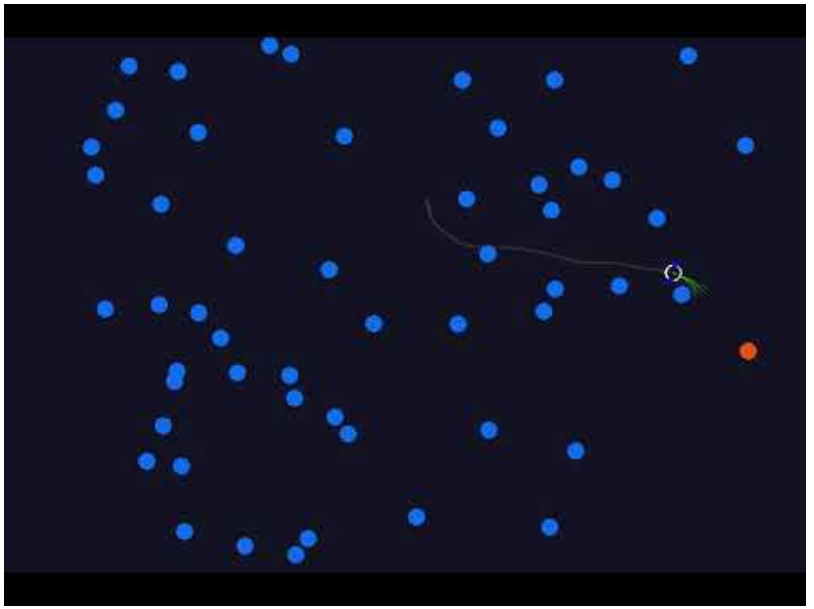


- Global & Local Planner

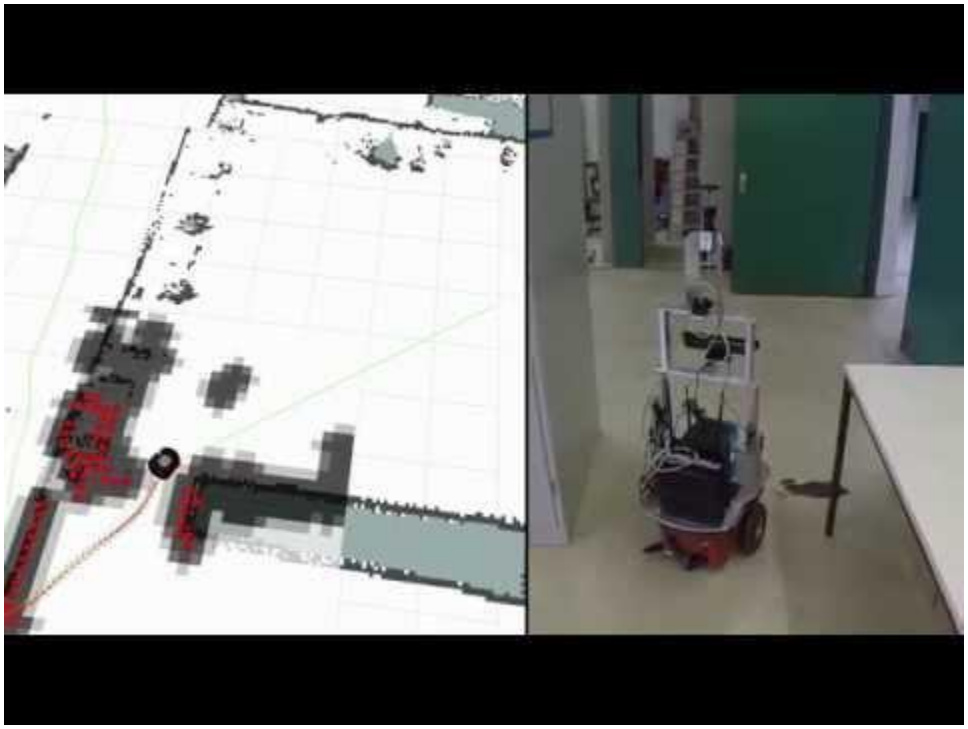
Local Planner	Global Planner
Sensor-based	Map-based
Reactive Navigation	Deliberative Navigation
Fast Response	Relatively Slower Response
Workspace area is incomplete or partially incomplete	Workspace area is known
Generate the path and moving toward target while avoiding obstacles or objects	Generate a feasible path before moving toward the goal position
Done online	Done offline

- Local Planner
- Dynamic Window Approach
- Sampling을 통해 다양한 선속도 및 회전 속도를 생성하고, 이를 기반으로 시뮬레이션을 진행하여, 점수 측정을 통해 높은 점수를 얻은 경로로 주행하는 방식

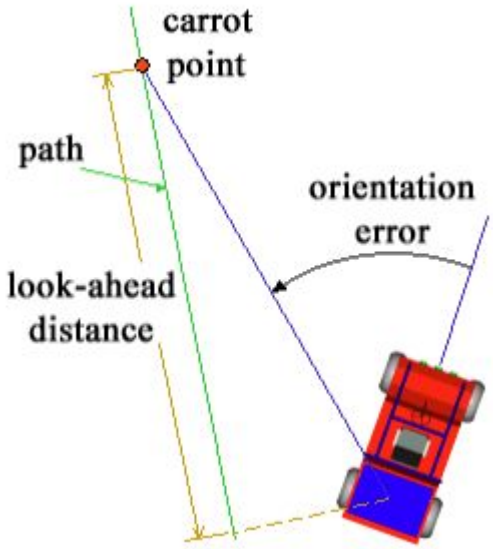


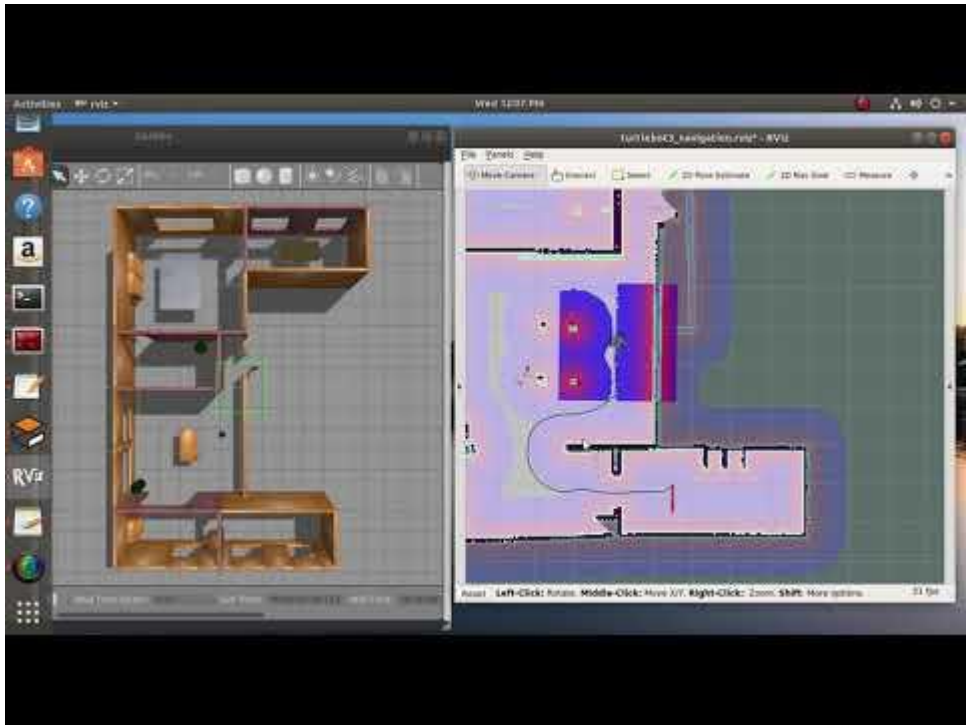


- Local Planner
- Time Elastic Band Planner
- 시간, 장애물, 역학적인 제한(Steering angle 등)을 고려하여 최적의 경로를 생성하는 알고리즘.



- Local Planner
- Active Scene Recognition Local Planner
- Follow the Carrot과 유사한 형태의 Planner
- Carrot Point를 지정하여, Orientation Error를 최소화하는 Controller.





02

Path Planning Package

WeGo

ROS Move Base package

- **move_base - planning & control the robot**
- **Published Topics**
  - cmd_vel (geometry_msgs/Twist) - A stream of velocity commands meant for execution by a mobile base.
- **Subscribed Topics**
  - move_base_simple/goal (geometry_msgs/PoseStamped) - Provides a non-action interface to move_base for users that don't care about tracking the execution status of their goals.
- **Parameters(Default)**
  - ~base_global_planner ("navfn/NavfnROS") - The name of the plugin for the global planner to use with move_base, see pluginlib documentation for more details on plugins. This plugin must adhere to the nav_core::BaseGlobalPlanner interface specified in the nav_core package.
  - ~base_local_planner ("base_local_planner/TrajectoryPlannerROS") - The name of the plugin for the local planner to use with move_base see pluginlib documentation for more details on plugins. This plugin must adhere to the nav_core::BaseLocalPlanner interface specified in the nav_core package.
  - ~recovery_behaviors ([{name: conservative_reset, type: clear_costmap_recovery/ClearCostmapRecovery}, {name: rotate_recovery, type: rotate_recovery/RotateRecovery}, {name: aggressive_reset, type: clear_costmap_recovery/ClearCostmapRecovery}] - A list of recovery behavior plugins to use with move_base, see pluginlib documentation for more details on plugins. These behaviors will be run when move_base fails to find a valid plan in the order that they are specified. After each behavior completes, move_base will attempt to make a plan. If planning is successful, move_base will continue normal operation. Otherwise, the next recovery behavior in the list will be executed. These plugins must adhere to the nav_core::RecoveryBehavior interface specified in the nav_core package.
  - ~controller_frequency (20.0) - The rate in Hz at which to run the control loop and send velocity commands to the base.

ROS Move Base package

- **move_base - planning & control the robot**
- **Parameters(Default)**
  - ~planner_patience (5.0) - How long the planner will wait in seconds in an attempt to find a valid plan before space-clearing.
  - ~controller_patience (15.0) - How long the controller will wait in seconds without receiving a valid control before space-clearing.
  - ~conservative_reset_dist (3.0) - The distance away from the robot in meters beyond which obstacles will be cleared from the costmap when attempting to clear space in the map. Note, this parameter is only used when the default recovery behaviors are used for move_base
  - ~recovery_behavior_enabled (true) - Whether or not to enable the move_base recovery behaviors to attempt to clear out space.
  - ~clearing_rotation_allowed (true) - Determines whether or not the robot will attempt an in-place rotation when attempting to clear space.
  - ~shutdown_costmaps (false) - Determines whether or not to shutdown the costmaps of the node when move_base is in an inactive state
  - ~oscillation_timeout (0.0) - How long in seconds to allow for oscillation before executing recovery behaviors. A value of 0.0 corresponds to an infinite timeout.
  - ~oscillation_distance (0.5) - How far in meters the robot must move to be considered not to be oscillating. Moving this far resets the timer counting up to the ~oscillation_timeout
  - ~planner_frequency (0.0) - The rate in Hz at which to run the global planning loop. If the frequency is set to 0.0, the global planner will only run when a new goal is received or the local planner reports that its path is blocked
  - ~max_planning_retries (-1) - How many times to allow for planning retries before executing recovery behaviors. A value of -1.0 corresponds to an infinite retries.
- http://wiki.ros.org/move_base

WeGo

ROS Move Base package

- 참고 자료
- https://github.com/Git-Hub-Pro/ROS_Navigation_in_5days
- https://atsushisakai.github.io/PythonRobotics/
- https://pythonrobotics.readthedocs.io/en/latest/modules/path_planning.html#cubic-spline-planning
- https://github.com/AtsushiSakai/PythonRobotics
-

WeGo

Q & A

go.support@wego-robotics.com

go.sales@wego-robotics.com